Assignment1:
Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and card
up to the third normal form.

Ans:

ER Diagram:

Here is the ER diagram for the given business scenario:

Entities:

Student
Attributes:
---->Student ID (primary key)
---->Name
---->Email
---->Password
Course
Attributes:
---->ID (primary key)
---->Course Name
---->Credits
---->Description
Instructor
Attributes:
---->Instructor ID (primary key)
---->Name
---->Email
---->Department
Registration
Attributes:
---->Registration ID (primary key)
---->Student ID (foreign key referencing Student)
---->Course ID (foreign key referencing Course)
---->Registration Date
---->Grade (initially null)
Grade
Attributes:
--->Grade ID (primary key)
--->Registration ID (foreign key referencing Registration)
--->Grade (e.g., A, B, C, etc.)

Relationships:

->A student can register for multiple courses (one-to-many).
Student -> Registration (one-to-many)
->A course can have multiple students registered (one-to-many).
Course -> Registration (one-to-many)
->An instructor can teach multiple courses (one-to-many).
Instructor -> Course (one-to-many)
->A registration is associated with one student and one course (many-to-one).
Registration -> Student (many-to-one)
Registration -> Course (many-to-one)
->A grade is associated with one registration (many-to-one).
Grade -> Registration (many-to-one)

Cardinality:

Student -> Registration: 1:N
Course -> Registration: 1:N
Instructor -> Course: 1:N
Registration -> Student: N:1
Registration -> Course: N:1
Grade -> Registration: N:1
Normalization:

The ER diagram is normalized up to the third normal form (3NF) as follows:

First Normal Form (1NF): Each table cell contains a single value.
Second Normal Form (2NF): Each non-key attribute in a table depends on the entire primary key.
Third Normal Form (3NF): If a table is in 2NF, and a non-key attribute depends on another non-key attribute, then it s
==========
Assignment2:
Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and
s between tables.

Ans:

Library System Database Schema:

1.Table: Author

->Fields:
  --->Author ID (Primary Key, Auto-increment, Integer, NOT NULL)
  --->First Name (VARCHAR(50), NOT NULL)
  --->Last Name (VARCHAR(50), NOT NULL)
  --->Birth Date (DATE, NOT NULL)
  --->Death Date (DATE, NULL)
  --->Biography (TEXT, NULL)
->Constraints:
  --->UNIQUE (Author ID)
  --->CHECK (Birth Date <= Death Date)

2.Table: Book

->Fields:
  --->Book ID (Primary Key, Auto-increment, Integer, NOT NULL)
  --->Title (VARCHAR(100), NOT NULL)
  --->Publication Date (DATE, NOT NULL)
  --->Publisher (VARCHAR(50), NOT NULL)
  --->Pages (Integer, NOT NULL)
  --->Language (VARCHAR(20), NOT NULL)
  --->Author ID (Foreign Key referencing Author, Integer, NOT NULL)
->Constraints:
  --->UNIQUE (Book ID)
  --->CHECK (Publication Date <= CURDATE())

3.Table: Genre

->Fields:
  --->Genre ID (Primary Key, Auto-increment, Integer, NOT NULL)
  --->Genre Name (VARCHAR(50), NOT NULL)
->Constraints:
  -->UNIQUE (Genre ID)

4.Table: Book_Genre

->Fields:

   --->Book ID (Foreign Key referencing Book, Integer, NOT NULL)
   --->Genre ID (Foreign Key referencing Genre, Integer, NOT NULL)
->Constraints:
   --->PRIMARY KEY (Book ID, Genre ID)
   --->CHECK (Book ID > 0 AND Genre ID > 0)

5.Table: Member

->Fields:
   --->Member ID (Primary Key, Auto-increment, Integer, NOT NULL)
   --->First Name (VARCHAR(50), NOT NULL)
   --->Last Name (VARCHAR(50), NOT NULL)
   --->Email (VARCHAR(50), NOT NULL, UNIQUE)
   --->Phone Number (VARCHAR(15), NOT NULL, UNIQUE)
   --->Address (VARCHAR(100), NOT NULL)
   --->City (VARCHAR(50), NOT NULL)
   --->State (VARCHAR(20), NOT NULL)
   --->Zip Code (VARCHAR(10), NOT NULL)
->Constraints:
   --->UNIQUE (Email)
   --->UNIQUE (Phone Number)

6.Table: Loan

->Fields:
   --->Loan ID (Primary Key, Auto-increment, Integer, NOT NULL)
   --->Book ID (Foreign Key referencing Book, Integer, NOT NULL)
   --->Member ID (Foreign Key referencing Member, Integer, NOT NULL)
   --->Checkout Date (DATE, NOT NULL)
   --->Due Date (DATE, NOT NULL)
   --->Return Date (DATE, NULL)
->Constraints:
   --->CHECK (Checkout Date <= Due Date)
   ---->CHECK (Return Date IS NULL OR Return Date >= Due Date)
==========
Assignment3:
Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that
ow concurrency control.

Ans:
ACID Properties of a Transaction:

A transaction is a sequence of operations that are executed as a single, all-or-nothing unit of work. The ACID propert
d securely. Here's what each property means in simple terms:

-->Atomicity: A transaction is treated as a single, indivisible unit. If any part of the transaction fails, the entire transac
s previous state.
-->Consistency: A transaction must leave the database in a consistent state, ensuring that data integrity is maintaine
nstraints defined in the database.
-->Isolation: A transaction should not be affected by other transactions that are running concurrently. This means th
erence from other transactions.
-->Durability: Once a transaction is committed, its effects are permanent and cannot be rolled back. The database en
 in the event of a system failure.

Transaction with Locking:

START TRANSACTION;

-- Lock the accounts to prevent concurrent access
SELECT * FROM accounts WHERE account_id = 1 FOR UPDATE;

```
SELECT * FROM accounts WHERE account_id = 2 FOR UPDATE;

-- Debit from account 1
UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;

-- Credit to account 2
UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;

-- Commit the transaction
COMMIT;
```

Serializable Isolation Level:

```
-- T1: Set isolation level to SERIALIZABLE
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

-- Read the initial balance of account 1
SELECT balance FROM accounts WHERE account_id = 1;

-- Wait for 5 seconds to allow T2 to run
SLEEP(5);

-- Read the balance of account 1 again
SELECT balance FROM accounts WHERE account_id = 1;

COMMIT;
```

==========

Assignment4:

Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use
s to remove a redundant table.

Ans:

Create Database:

```
-->CREATE DATABASE LibrarySystem;
```

Create Tables:

```
-->USE LibrarySystem;

-CREATE TABLE Authors (
-  AuthorID INT PRIMARY KEY AUTO_INCREMENT,
-  FirstName VARCHAR(50) NOT NULL,
-  LastName VARCHAR(50) NOT NULL,
-  BirthDate DATE NOT NULL,
-  DeathDate DATE,
-  Biography TEXT
-);

-CREATE TABLE Books (
-  BookID INT PRIMARY KEY AUTO_INCREMENT,
-  Title VARCHAR(100) NOT NULL,
-  PublicationDate DATE NOT NULL,
-  Publisher VARCHAR(50) NOT NULL,
-  Pages INT NOT NULL,
-  Language VARCHAR(20) NOT NULL,
-  AuthorID INT NOT NULL,
-  FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
-);

-CREATE TABLE Genres (
-  GenreID INT PRIMARY KEY AUTO_INCREMENT,
```

- GenreName VARCHAR(50) NOT NULL
-);

-CREATE TABLE Book_Genres (
- BookID INT NOT NULL,
- GenreID INT NOT NULL,
- PRIMARY KEY (BookID, GenreID),
- FOREIGN KEY (BookID) REFERENCES Books(BookID),
- FOREIGN KEY (GenreID) REFERENCES Genres(GenreID)
-);

-CREATE TABLE Members (
- MemberID INT PRIMARY KEY AUTO_INCREMENT,
- FirstName VARCHAR(50) NOT NULL,
- LastName VARCHAR(50) NOT NULL,
- Email VARCHAR(50) NOT NULL UNIQUE,
- PhoneNumber VARCHAR(15) NOT NULL UNIQUE,
- Address VARCHAR(100) NOT NULL,
- City VARCHAR(50) NOT NULL,
- State VARCHAR(20) NOT NULL,
- ZipCode VARCHAR(10) NOT NULL
-);

-CREATE TABLE Loans (
- LoanID INT PRIMARY KEY AUTO_INCREMENT,
- BookID INT NOT NULL,
- MemberID INT NOT NULL,
- CheckoutDate DATE NOT NULL,
- DueDate DATE NOT NULL,
- ReturnDate DATE,
- FOREIGN KEY (BookID) REFERENCES Books(BookID),
- FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
-);
Modify Table Structures:
-ALTER TABLE Books
-ADD COLUMN ISBN VARCHAR(20) NOT NULL;

-ALTER TABLE Loans
-ADD COLUMN RenewalCount INT NOT NULL DEFAULT 0;
Drop Redundant Table:
-->DROP TABLE Book_Genres;
==========
Assignment5:
Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDE
on.

Ans:
Create Index:
->USE LibrarySystem;
CREATE INDEX idx_Title ON Books (Title);

Query to Demonstrate Index Performance:
->SELECT * FROM Books WHERE Title = 'To Kill a Mockingbird';

Drop Index:
->DROP INDEX idx_Title ON Books;
==========
Assignment6:
Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a scr

Ans:
Create a New Database User:
->CREATE USER 'library_user'@'localhost' IDENTIFIED BY 'password';

Grant Privileges:
->GRANT SELECT, INSERT, UPDATE, DELETE ON LibrarySystem.* TO 'library_user'@'localhost';

Revoke Privileges:
->REVOKE UPDATE, DELETE ON LibrarySystem.* FROM 'library_user'@'localhost';

Drop the User:
->DROP USER 'library_user'@'localhost';
==========
Assignment7:
Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new
e BULK INSERT operations to load data from an external source.

Ans:
INSERT:
-- Insert a new author
INSERT INTO Authors (AuthorID, FirstName, LastName, BirthDate, DeathDate)
VALUES (10, 'John', 'Doe', '1970-01-01', NULL);

-- Insert a new book
INSERT INTO Books (BookID, Title, AuthorID, Publisher, PublicationDate, ISBN)
VALUES (100, 'New Book', 10, 'New Publisher', '2020-01-01', '1234567890');

-- Insert a new borrower
INSERT INTO Borrowers (BorrowerID, FirstName, LastName, Email, Phone)
VALUES (1000, 'Jane', 'Doe', 'jane.doe@example.com', '123-456-7890');

-- Insert a new loan
INSERT INTO Loans (LoanID, BookID, BorrowerID, LoanDate, DueDate)
VALUES (10000, 100, 1000, '2022-01-01', '2022-01-31');

UPDATE:
-- Update an author's birth date
UPDATE Authors
SET BirthDate = '1965-01-01'
WHERE AuthorID = 10;

-- Update a book's publication date
UPDATE Books
SET PublicationDate = '2021-01-01'
WHERE BookID = 100;

-- Update a borrower's email
UPDATE Borrowers
SET Email = 'jane.doe2@example.com'
WHERE BorrowerID = 1000;

DELETE Operations:
-- Delete a book
DELETE FROM Books
WHERE BookID = 100;

-- Delete a borrower
DELETE FROM Borrowers
WHERE BorrowerID = 1000;

```
-- Delete an author
DELETE FROM Authors
WHERE AuthorID = 10;

BULK INSERT :
BULK INSERT Authors
FROM 'C:\Path\To\authors.csv'
WITH
(
   FORMATFILE = 'C:\Path\To\authors.xml',
   FIRSTROW = 2,
   IGNOREBLANKROWS = 1
);
```